Internal Ptrs

Data request object code

Oct 6, 1989

When a data request is initialized, the array of idents for each listype in the request is "compiled" into an array of Internal Ptrs in order to more efficiently update the answers. This translation is done even for a one-shot request.

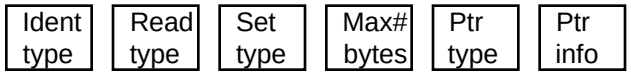
There are two variations of data requests. The first is a data request that originates from a Local Station (either for the current application page or in response to a Data Server request over the network). Such a request may include data from various nodes; the local station selects out the non-local idents from the request to form an "external request" that is sent to the network. Each node receiving the network request responds to its part of the request only, and the various "answer fragments" are put together in original request order to build the response to the original request. The module that processes such a request is REQD, and the module that includes the "compiling loops" is REQDGENP, because it generates the internal ptrs for an REQD-style request. The REQD module is the "odds-on favorite" for having the most complicated logic in the system code. REQDGENP is a subroutine called by REQD and is therefore an extension of REQD. As such, it inherits much of REQD's complexity.

The other variation of data request is the "ordinary" network data request. In this case, the request may include references to various nodes, but the processing of the request will result in building only the local node's answer fragment. (Obviously, it is this ordinary network request that is used when a node processes the first data request variation.) The ordinary network request is processed by the PREQD module which calls PREQDGEN as a subroutine. The code for this version of the "compiling loops" is simpler than that in REQDGENP, because references to data from other nodes can be ignored, and an external request does not have to be built.

Since there are two variations of data requests processed by two different routines (REQDATA and PREQDATA), which in turn invoke two different subroutines (REQDGENP and PREQDGEN), it is necessary to write two different "compiling loops" when a new type of internal ptr must be generated.

Listype table

The format of a listype entry is as follows:



The *ident type* byte is a small index into the ident table the gives the (short) length of the ident. It is also used to insure that a request which specifies more than one listype uses only listypes that are ident-compatible, which just means that all listypes in the request use the same ident type.

The *read type* byte is an index into the READS branch table in the COLLECT module and thus selects the read type routine that produces the answers to a data request given the array of internal ptrs corresponding to the array of idents in the original request.

The *set type* byte is an index into the SETS branch table in the SETDATA module and thus selects the set type routine that accomplishes a setting action given the ident and the accompanying data. The *max#* byte is the maximum number of bytes of setting data that are acceptable for that listype.

The *ptr type* byte is of two varieties. If its value is < 32, then it is a table#, and the *ptr info* byte is an offset to a field of an entry; if it is ≥ 32, the ptr type byte is used to index into the GENS table in the appropriate module (REQDGENP or PREQDGEN) to select the ptr type routine that produces an array of internal ptrs given an array of idents.

REQDGENP compiling loop

The register-based call to the ptr type routine is as follows:

D2.L= Ext ident cntr in hi word, #idents (bit#15 = long id flag) in lo word
D4.W= Ptr info byte
D5.B= local node#
D6.W= #bytes of data requested from each ident
A1.L= Ptr to array of idents
A2.L= Ptr to output array of internal ptrs
A3.L= Ptr into array of external idents

Upon exit, A2 should be advanced reflecting the number of internal ptrs that have been stored. The number of external idents is incremented in the upper word of D2 as they are encountered. For each such ident, the ident itself is copied into the array of external idents pointed to by A3. This is used in the external data request that will be sent to the network to collect the answer fragments from the other node(s) referenced in the request. Other registers are scratch except, of course, A5/A6.

PREQDGEN compiling loop

This case is simpler from the previous case since all idents which reference data from other nodes can be ignored. The register-based call to the ptr type routine is as follows:

D2.W= #idents (bit#15 = long ident flag)
D4.W= Ptr info byte
D5.B= local node#
D6.W= #bytes of data requested from each ident
A1.L= Ptr to array of idents
A2.L= Ptr to output array of internal ptrs

Upon exit, A2 should be advanced reflecting the number of internal ptrs that have been stored. Other registers are scratch except A3/A5/A6.

Internal ptr types

Ptr type < 32. Table entry field ptr

If entry# out-of-range, use ptr to zeros, else use ptr to field in table entry.

Ptr type = 32. Memory address ident

For short ident case, map 24 bits into 32 bits by using `$FF` for the hi byte if the 24-bit value is ≥ `$F00000`, else use `$00` for the hi byte. For the long ident case, use the full 32-bit address for the internal ptr.

Ptr type = 33. Device name to channel ident

There is no long ident case, as the ident is merely the 6-character name. The internal ptr is the node# in the hi word and the channel# in the lo word.

Ptr type = 34. Binary status via Bit ident

Use the lo 24 bits of the internal ptr to store the address of the `BBYTE` entry which contains the value of the status byte. The hi 8 bits contains the bit# in the range 0–7 of the bit in the byte to be sampled.

Ptr type = 35. Global variables (relative to A5)

The internal ptr is merely the address of the global variable.

Ptr type = 36. Generally interesting data

The internal ptr is merely the address of the byte in the `G.I.D.` pool.

Ptr type = 37. Serial input queue data

The internal ptr is the base address of the serial input queue. If there were more than one serial port, it would be logical to use the base address of the serial input queue associated with the port according to the ident value.

Ptr type = 38. Data stream data

The data stream index is in the hi word and the ptr info is in the lo word. The ptr info (and hence the listype#) can be used to identify whether old data is being included in the request.

External answer buffer ptrs

For the local or data server types of requests, the internal ptr may be marked as one which points into an external answer buffer. Normally, bit #31 is used for this purpose. The read type routine, when it sees the sign bit set, will simply copy the bytes of memory pointed to by the other 31 bits; any special processing to produce the data was already done by the node that sent the answer fragment that is copied by the Network Task into the external answer buffer. In the case of a memory address used by read types #1 or #4, where the job of the read routine is merely to copy the memory pointed to by the address, this special use of bit #31 is not used, as the job is to copy memory bytes in any case, no matter what it points to.

For writing new ptr type routines, it is advised to study some of the seven current examples for ptr types 32–38 above. Recall that two routines should be written, one referenced from `REQDGENP` and the other from `PREQDGEN`. The latter type is simpler than the former.